

02. 7. 2004

日 本 国 特 許 庁
JAPAN PATENT OFFICE

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office.

出 願 年 月 日
Date of Application: 2 0 0 3 年 6 月 2 7 日

出 願 番 号
Application Number: 特 願 2 0 0 3 - 1 8 5 4 8 1
[ST. 10/C]: [J P 2 0 0 3 - 1 8 5 4 8 1]

REC'D. 26 AUG 2004

WIPO PCT

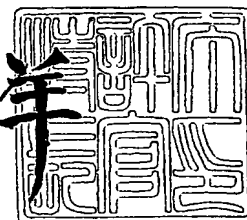
出 願 人
Applicant(s): アイピーフレックス株式会社

PRIORITY DOCUMENT
SUBMITTED OR TRANSMITTED IN
COMPLIANCE WITH
RULE 17.1(a) OR (b)

2 0 0 4 年 8 月 1 2 日

特許庁長官
Commissioner,
Japan Patent Office

小 川 洋



【書類名】 特許願

【整理番号】 030136P502

【あて先】 特許庁長官殿

【国際特許分類】 G06F 9/45

【発明者】

 【住所又は居所】 東京都品川区上大崎二丁目 2 7 番 1 号 アイピーフレックス株式会社内

 【氏名】 志村 大

【特許出願人】

 【識別番号】 500238789

 【氏名又は名称】 アイピーフレックス株式会社

【代理人】

 【識別番号】 100102934

 【弁理士】

 【氏名又は名称】 今井 彰

【手数料の表示】

 【予納台帳番号】 050728

 【納付金額】 21,000円

【提出物件の目録】

 【物件名】 明細書 1

 【物件名】 図面 1

 【物件名】 要約書 1

【ブルーフの要否】 要

【書類名】 明細書

【発明の名称】 並列処理システムの生成方法

【特許請求の範囲】

【請求項 1】 並列に動作する複数の要素を備えたシステムにより、同期して独立に行われる複数の並列処理をそれぞれ規定した複数の並列記述を有する定義ファイルであって、前記複数の並列記述は、少なくとも 1 つの他の並列処理の出力データを少なくとも 1 つの入力データとする並列処理を示す並列記述を含み、複数の入力データを備えた前記並列記述においては、それら複数の入力データは、当該システムに入力されてからのレイテンシーが同一である定義ファイルが記録されているコンピュータ読み取り可能な記録媒体。

【請求項 2】 請求項 1 において、前記複数の並列記述は、前記複数の要素が動作するクロックに同期して実行される前記複数の並列処理をそれぞれ規定する、コンピュータ読み取り可能な記録媒体。

【請求項 3】 独立に行われる複数の並列処理をそれぞれ規定した複数の並列記述を有する定義ファイルに基づき、並列に動作する複数種類の要素を備えた並列処理システムを生成する方法であって、

前記定義ファイルの複数の並列記述には、少なくとも 1 つの他の並列処理の出力データを少なくとも 1 つの入力データとする並列処理を示す並列記述が含まれており、

前記複数種類の要素の情報を記録したハードウェアライブラリに基づき、前記並列処理を実行するための回路構成であって、少なくとも 1 つの前記要素を含む回路構成を生成する第 1 の工程と、

前記回路構成に複数のデータが入力される場合は、それら複数の入力データの、当該並列処理システムに入力されてからのレイテンシーが一致するように遅延要素を加える第 2 の工程とを有する並列処理システムの生成方法。

【請求項 4】 請求項 3 において、前記並列処理システムは、前記複数種類の要素の接続を変えることにより異なるハードウェア構成を実現可能であり、

前記第 2 の工程により生成された前記回路構成を含む前記ハードウェア構成を前記複数種類の要素の接続情報を含むハードウェア構成情報として出力する工程

をさらに有する並列処理システムの生成方法。

【請求項 5】 請求項 3 において、前記複数種類の要素は、単体で前記並列処理を処理可能な程度の規模の複数種類の演算ユニットである、並列処理システムの生成方法。

【請求項 6】 請求項 3 において、前記複数種類の要素は、バイトあるいはワード単位で異なった演算を実行可能な複数種類の演算ユニットである、並列処理システムの生成方法。

【請求項 7】 請求項 3 において、前記第 2 の工程では、前記複数の要素が動作するクロックに同期した前記遅延要素を加える、並列処理システムの生成方法。

【請求項 8】 請求項 3 において、前記並列記述により規定された並列処理が、他の並列記述により規定された並列処理の少なくとも一部と同じ共通処理を含む場合は、前記第 1 の工程では、前記共通処理に同一の前記要素を含む共通構成を割り当て、前記第 2 の工程では、前記共通構成に含まれない前記要素に対して遅延要素を加える、並列処理システムの生成方法。

【請求項 9】 独立に行われる複数の並列処理をそれぞれ規定した複数の並列記述を有する定義ファイルに基づき、並列に動作する複数種類の要素を備えた並列処理システムを生成する生成装置であって、

前記定義ファイルの複数の並列記述には、少なくとも 1 つの他の並列処理の出力データを少なくとも 1 つの入力データとする並列処理を示す並列記述が含まれており、

前記複数種類の要素の情報を記録したハードウェアライブラリに基づき、前記並列処理を実行するための回路構成であって、少なくとも 1 つの前記要素を含む回路構成を生成する第 1 の手段と、

前記回路構成に複数のデータが入力される場合は、それら複数の入力データの、当該並列処理システムに入力されてからのレイテンシーが一致するように遅延要素を加える第 2 の手段とを有する並列処理システムの生成装置。

【請求項 10】 請求項 9 において、前記並列処理システムは、前記複数種類の要素の接続を変えることにより異なるハードウェア構成を実現可能であり、

前記第2の手段により生成された前記回路構成を含む前記ハードウェア構成を前記複数種類の要素の接続情報を含むハードウェア構成情報として出力する手段をさらに有する並列処理システムの生成装置。

【請求項11】 独立に行われる複数の並列処理をそれぞれ規定した複数の並列記述を有する定義ファイルに基づき、並列に動作する複数種類の要素を備えたシステムを設計する設計プロセスをコンピュータにより実行可能とするプログラムであって、

前記定義ファイルの複数の並列記述には、少なくとも1つの他の並列処理の出力データを少なくとも1つの入力データとする並列処理を示す並列記述が含まれており、

前記複数種類の要素の情報を記録したハードウェアライブラリに基づき、前記並列処理を実行するための回路構成であって、少なくとも1つの前記要素を含む回路構成を生成する第1の工程と、

前記回路構成に複数のデータが入力される場合は、それら複数の入力データの、当該並列処理システムに入力されてからのレイテンシーが一致するように遅延要素を加える第2の工程とを有するプログラム。

【請求項12】 独立に行われる複数の並列処理をそれぞれ規定した複数の並列記述を有する定義ファイルに基づき、並列に動作する複数種類の要素を備えたシステムをシミュレートする方法であって、

前記定義ファイルの複数の並列記述には、少なくとも1つの他の並列処理の出力データを少なくとも1つの入力データとする並列処理を示す並列記述が含まれており、

前記定義ファイルに含まれる複数の並行処理を同期して実行する工程を有し、この実行する工程では、複数の入力データを備えた前記並列記述を実行する際には、それら複数の入力データに、当該システムに入力されてからのレイテンシーが同一のデータを用いる、シミュレーション方法。

【請求項13】 独立に行われる複数の並列処理をそれぞれ規定した複数の並列記述を有する定義ファイルに基づき、並列に動作する複数種類の要素を備えたシステムをシミュレートするシミュレータであって、

前記定義ファイルの複数の並列記述には、少なくとも 1 つの他の並列処理の出力データを少なくとも 1 つの入力データとする並列処理を示す並列記述が含まれており、

前記定義ファイルに含まれる複数の並行処理を同期して実行する手段を有し、この実行する手段では、複数の入力データを備えた前記並列記述を実行する際に、それら複数の入力データに、当該システムに入力されてからのレイテンシーが同一のデータを用いる、シミュレータ。

【請求項 14】 独立に行われる複数の並列処理をそれぞれ規定した複数の並列記述を有する定義ファイルに基づき、並列に動作する複数種類の要素を備えたシステムをコンピュータによりシミュレートするプログラムであって、

前記定義ファイルの複数の並列記述には、少なくとも 1 つの他の並列処理の出力データを少なくとも 1 つの入力データとする並列処理を示す並列記述が含まれており、

前記定義ファイルに含まれる複数の並行処理を同期して実行する工程を有し、この実行する工程では、複数の入力データを備えた前記並列記述を実行する際に、それら複数の入力データに、当該システムに入力されてからのレイテンシーが同一のデータを用いる、プログラム。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】

本発明は、並列処理システムの設計に関するものである。

【0002】

【従来の技術】

L S I あるいは A S I C を設計するために幾つかの言語が用いられている。C 言語などの抽象度の高い高級言語と称されるものは、手続き的レベルの言語であり、1 命令毎にいかに関理全体が順番に実行されていくかを示すのに適している。このレベルの記述は、一般的にハードウェア依存性がなく、適当なコンピュータで受け入れられるアプリケーションプログラムであり、L S I の仕様、または L S I において実行する処理全体を一般的に記述するために用いられる。V e r

ilog-HDLあるいはVHDLなどのハードウェア記述言語（HDL）は、RTLと称されることもありレジスタランジスタレベルで、特定のハードウェアにより特定の命令が実行されているデータパスやパスを駆動するシーケンスを記述するために用いられる。

【0003】

【特許文献1】

特開平10-116302号公報

【0004】

【発明が解決しようとする課題】

アルゴリズムは、問題を解くために明確に規定され、順序付けられた有限個の規則からなる集合として定義されており、従来、並列処理は、アルゴリズムにより記述された処理全体（アプリケーション）を順番に進める上で、独立して実行可能な部分（処理）を並列に実行し、処理時間を短縮する目的で用いられる。予め並列処理に適したハードウェアリソースを備えたシステムでアプリケーションを実行する場合は、コンパイラなどにより並列処理ができる部分は並列化され、実行速度を向上することがトライされる。

【0005】

また、特定のアプリケーションの実行を目的としたハードウェアを設計する場合は、独立して実行できる部分を並列に処理するように回路を設計して処理時間の短縮を目指す。特開平10-116302号公報に記載された技術は、並列処理および同期通信等により実行時間が定数でない処理を記述可能なHDLにより回路を設計する方法である。同期通信とは、2つのファンクションを並行に実行する際に、それらに含まれるプロセスを送信側が準備できるまで受信側が待ち、通信が完了した後にプロセスが進行する。したがって、これらのファンクションは並列に記述されても独立して実行されず、実行時間が可変の処理となる。一方、同期通信を行わない処理は並列処理として独立して行われる。これらは、ソース言語で与えられた処理全体の内、ソース上では並列に実行するように記載された処理を、ハードウェア設計において並列に、あるいは同期通信を用いて実行して実行サイクル数を短縮することを目的とした技術である。

【0006】

近年、LSIを構成する回路の一部をソフトウェアにより再構成できるハードウェアが提供されている。さらに、国際公開WO03/007155号には、再構成する基本単位を、ゲートレベルからALUなどのある程度の規模の演算機能を備えた演算ユニットにして、複数種類の演算ユニットをマトリクス状に配置し、再構成に要する時間を短縮することが開示されている。複数の演算ユニットがマトリクス状に配置されたシステムは、それぞれの演算ユニットが並列に処理を実行できるので、膨大な数の並列処理に適したハードウェアリソースを備えたシステムと捉えることが可能である。しかしながら、この種の並列処理に適したシステムを設計するのに適した設計システムは提供されていない。

【0007】

C言語などのソフトウェア設計に適した高級言語は、アルゴリズムを、それに含まれる規則を時間的な順番に処理することを前提としている。したがって、プログラムカウンタを進めて命令がシーケンシャルに実行されるように構成されており、シーケンシャルではない並列という概念を導入することは難しい。命令を並列的に記述することが許容されたとしても、それは、時間的な順番に齟齬をきたさない範囲で、独立して実行できる処理を空間的に並列に広げて実行できる程度であり、並列処理に適したハードウェアリソースを積極的に使用することはできない。さらに、高級言語であれば、ハードウェアに依存しない命令が記述されるために、並列に記述した命令がハードウェアで実際に開始されたり、終了したりするタイミングは不明である。したがって、処理できる範囲を空間的に広げても、設計者はハードウェア上で実際にどのように並列処理が実行されるのかは定義できないし、把握することもできない。

【0008】

HDLは独立して動作する回路構成を記述するので、本来、並列処理を記述するものである。また、ハードウェアが明確になるので、処理が実行されるタイミングを調べたり、調整できる。このため、高級言語で与えられたアルゴリズムを実現するHDLを記述することができる。しかしながら、逆に、特定のハードウェアを前提として記述されるので、汎用性はなく、ハードウェアが異なれば同一

のアルゴリズムを実現することは不可能である。また、HDLが対象としているハードウェアが分からなければHDLに含まれているアルゴリズムを理解することもできない。

【0009】

高級言語を特定のハードウェア用にコンパイルした結果として得られるマイクロプログラムのレベルでは、完全に独立して実行可能な命令を並列に記述するVLIW技術、複数の命令を同時にフェッチして並列実行できる命令を見つけ出して実行するスーパースケラ技術がある。これらは時間的に並んだ処理のうち、それを実行するために用意された複数のパイプラインで空間的に並列に実行できるものを並列に処理して実行速度を改善する技術であり、時間的な順番に齟齬がおきない範囲を空間的に広げる点では高級言語と変わりはない。すなわち、マイクロプログラムもプログラム言語であり、プログラムカウンタを進めて命令を順番に実行するシーケンシャルな処理はVLIWやスーパースケラにおいても同様に必要とされる。さらに、マイクロプログラムは、特定のハードウェアを前提としており、HDLと同様に汎用性は少ない。

【0010】

このように、プログラムカウンタを進めて実行するプログラム言語では、高級言語であれば汎用性があり、ソフトウェア設計は行い易いが、並列処理に展開できる部分はプログラムカウンタを進めても独立して実行できる処理に限られてしまい、多数の並列処理に適した演算ユニットを有効に利用することは難しい。ハードウェア依存のない高級言語では、さらに、並列処理のタイミングも分からないので、マトリクス状に配置された多数の演算ユニットを並列に動作させてアプリケーションを効果的に実行する設計は不可能である。一方、HDLは並列処理の記述は可能であるが、アルゴリズムを記述したとしても特定のハードウェアを前提としてアルゴリズムが記述できるだけなので、特定のハードウェアの知識が必要である。したがって、マトリクス状に配置された多種類の演算ユニットの機能および入出力のタイミングなどをソフトウェア技術者が理解し、アプリケーションをHDLで設計することは無理がある。

【0011】

【課題を解決するための手段】

そこで、本発明においては、並列に動作する複数の要素を備えたシステムにより、同期して独立に行われる複数の並列処理をそれぞれ規定した複数の並列記述を有する定義ファイルであって、複数の並列記述は、少なくとも1つの他の並列処理の出力データを少なくとも1つの入力データとする並列処理を示す並列記述を含み、複数の入力データを備えた並列記述においては、それら複数の入力データは、当該システムに入力されてからのレイテンシーが同一である定義ファイルを提供する。この定義ファイルにより、ハードウェア依存性のない記述により、プログラムカウンタの必要のない形態でアルゴリズムを定義することができる。すなわち、この定義ファイルにより、従来的高级言語と類似した記述ではあるが、時間順序性のない並列記述により、アルゴリズムに含まれる順序を時間的にではなく空間的に記述できる。

【0012】

この定義ファイルは、処理させる仕事（アプリケーション）の手順を定義している点では、従来のプログラム言語と共通する。しかしながら、プログラム言語は、記述されたコンピュータに対する命令が、原則として最初から順番に、すなわち、時間的経過と共に実行されるものとして記述している。したがって、プログラム言語により定義されたアルゴリズムを実行するためには、実行する命令の順番を示すプログラムカウンタによる制御が必要となる。一方、記述された命令がプログラムカウンタにより順番に実行されるので、変数を含む命令を実行する際は、その命令に時間的に先行して実行される命令により変数の状態は一義的に決まり、定義されたアルゴリズムが不安定になることはない。

【0013】

一方、本発明の定義ファイルは、同期して独立に行われる複数の並列処理をそれぞれ規定した複数の並列記述を有する。したがって、本発明の定義ファイルに定義された並列処理を実行するには、時間的な順番を示すプログラムカウンタは必要としない。そして、他の並列処理の出力データを入力データとする並列処理を示す並列記述を含むことによりアルゴリズムを定義できる。しかしながら、変数を含む並列記述では、変数の経過が一義的に定まらなるとアルゴリズムが不安

定になる可能性がある。そこで、複数の入力データを備えた並列記述においては、それら複数の入力データは、当該システムに入力されてからのレイテンシーが同一であると解することにより、変数が不安定になることを防止している。したがって、本発明の定義ファイルでは、並列記述毎に変数が一義的に定まり、アルゴリズムを並列記述により正確に定義できる。また、本発明の定義ファイルでは、並列記述にしたがって並列処理は完全に独立して実行され、同期通信などの余分な処理がなくても、また、ハードウェアを特定することによりタイミング問題を解決しなくても、アルゴリズムを正確に記述できる。

【0014】

本明細書において、入力データのレイテンシーが同一とは、それらの入力データが、同期してシステムにロードされたデータ群またはその同期してロードされたデータ群のいずれかが少なくとも1つの並列処理により処理されたデータ群に含まれることを言う。すなわち、レイテンシーが同一の入力データは、システムに入力されたタイミングが同一のデータ、またはそのデータが加工されたものである。

【0015】

本発明の定義ファイルはHDLと同様に複数の並列記述からなり、複数の並列記述により並列処理を実行するためにプログラムカウンタが不要である。したがって、定義ファイルは、ハードウェアを記述していると言える。さらに、定義ファイルには、他の並列処理の出力データを入力データとする並列処理を示す並列記述を含み、複数の入力データがある場合もそれらの状態は一義的に決まる。したがって、本発明により、アルゴリズムを正確に、そして見て分かるように定義できるハードウェア記述言語を提供できる。さらに、ハードウェアの詳細が分からなくても入力データは一義的に決まるので、実際に並列処理を行うハードウェアの詳細な情報あるいは知識は不要である。したがって、本発明の定義ファイルは、実際のハードウェアに依存せず、特定のハードウェアを前提にしないでハードウェアを記述できる。このため、ハードウェアインディペンデントで、極めて汎用的なハードウェア記述言語である。したがって、ソフトウェア技術者が簡単にLSI、特に、並列処理要素を多数含んだLSIの設計あるいは生成するのに

適したツールあるいは記述となる。

【0016】

さらに、本発明の定義ファイルに含まれる複数の並列記述を、それらの入力データのレイテンシーの順番に並べると、複数の並列記述が、それに含まれる変数の時間経過の順番で並ぶことになる。この状態は、命令が実行される順番に並んだプログラムと同じである。したがって、ソフトウェア技術者は、普通にプログラムを作成するのと同じ感覚で、本発明の定義ファイルを作成することが可能である。この点でも、定義ファイルは、ソフトウェア技術者が簡単にLSI設計を行うのに正に適した設計あるいは生成するのに適したツールとなる。そして、本発明の定義ファイルにより、並列処理が可能な複数の要素により、アルゴリズムを空間的に割り付けることが可能となり、多数の並列に動作する要素を備えたシステムを有効に活用してアプリケーションを高速で実行できる。

【0017】

本発明の定義ファイルにおいては、並列記述毎に、入力データのレイテンシーが同一と判断され、タイミングが調整される。したがって、並列記述により規定される並列処理の単位で入力データのタイミングが調整されることになる。1つの並列処理が複数の演算素子、例えば、トランジスタなどで構成される回路で実行される場合は、タイミングの調整は並列処理単位より小さな単位、例えば、トランジスタ間で行うことが望ましいかもしれない。しかしながら、ALUなどのある程度の規模の演算機能を備えた演算ユニットがマトリクス状に配置されたシステムにおいては、並列記述に対して1つまたは数個の演算ユニットを割り当てることにより並列処理を実行できる。したがって、並列記述単位でタイミングを調整するためには、遅延用の演算ユニットを加えれば良く、本発明の定義ファイルは、複数種類の演算ユニットを接続してアプリケーションに適したハードウェア構成を実現する並列処理システムを記述するのに適している。このため、本発明の定義ファイルにより、並列処理システムを効率的に設計および開発することができ、また、シミュレートすることが可能となる。

【0018】

本発明の定義ファイルは、コンピュータ読み取り可能な記録媒体に記録して提

供することができ、定義ファイルに基づき、コンピュータを用いて、並列に動作する複数種類の要素を備えた並列処理システムを生成することができる。定義ファイルの複数の並列記述には、少なくとも1つの他の並列処理の出力データを少なくとも1つの入力データとする並列処理を示す並列記述が含まれている。したがって、複数種類の要素の情報を記録したハードウェアライブラリに基づき、並列処理を実行するための回路構成（ハードウェア構成）であって、少なくとも1つの要素を含む回路構成を生成する第1の工程と、並列記述が複数の入力データを備えており、並列処理を実行する回路構成に複数のデータが入力される場合は、それら複数の入力データの、並列処理システムに入力されてからのレイテンシーが一致するように遅延要素を加える第2の工程とにより、定義ファイルに定義されたアルゴリズムを複数種類の要素により空間的に割り付けでき、アプリケーションを実行する並列処理システムを生成できる。

【0019】

特定のハードウェアに関する情報が格納されたハードウェアライブラリにより各種種類の要素で消費される時間が分かる。したがって、並列記述が複数の入力データを備えている場合に、それらの複数の入力データのレイテンシーが一致するように、特定のハードウェアを前提とした遅延要素を加えることができ、特定のハードウェアに定義ファイルに定義されたアルゴリズムを正確に割付できる。複数の要素がクロックに同期して処理を実行するハードウェアであれば、ハードウェアライブラリには、複数種類の要素毎に消費されるサイクル数を含む情報が格納されており、第2の工程では、複数の要素が動作するクロックに同期した遅延要素を加えられる。

【0020】

並列処理システムは、ハードウェア構成が固定されるものであってもよい。並列処理システムは、複数種類の要素の接続を変えることにより異なるハードウェア構成を実現可能なものであってもよい。第2の工程または手段により生成されたハードウェア構成を複数種類の要素の接続情報を含むハードウェア構成情報として出力する工程を設けることにより、再構成用の情報が得られる。

【0021】

本発明の並列処理システムの生成方法は、並列に動作する複数種類の要素を備えたシステムを生成する並列処理システム生成装置、例えばコンパイラとしても提供することが可能である。このコンパイラは、定義ファイルに基づき、並列に動作する複数種類の要素を備えた並列処理システムを生成する生成装置であって、複数種類の要素の情報を記録したハードウェアライブラリに基づき、並列処理を実行するための回路構成であって、少なくとも1つの要素を含む回路構成を生成する第1の手段と、回路構成に複数のデータが入力される場合は、それら複数の入力データの、当該並列処理システムに入力されてからのレイテンシーが一致するように遅延要素を加える第2の手段とを有する。また、コンパイラとしての機能を、並列処理システム生成プログラムあるいはプログラム製品として適当な記録媒体に記録したり、コンピュータネットワークを用いて提供することが可能である。この並列処理システム生成用のコンパイラプログラムは、上述した第1の工程と、第2の工程とを実行可能な命令を有するプログラムである。

【0022】

上述したように、本発明の定義ファイルは、複数種類の要素が、単体で並列処理を処理可能な程度の規模の複数種類の演算ユニットである並列処理システムを生成するのに適している。したがって、ハードウェアライブラリには、単体で並列処理を処理可能な程度の規模の複数種類の演算ユニットの情報を用意することが望ましい。また、複数種類の要素はビット単位で演算するものであっても良い。しかしながら、定義ファイルに記述される並列処理は一般にバイトあるいはワード単位でのデータ処理である。したがって、ハードウェアライブラリには、バイトあるいはワード単位で異なった演算を実行可能な複数種類の演算ユニットの情報を用意し、複数の演算ユニットが配置された並列処理システムを生成することが望ましい。

【0023】

予め複数種類の演算ユニットがマトリクス状に配置され、演算ユニットを接続するネットワークあるいは回路配線の構成を変更することにより、異なるハードウェア構成にすることができる再構成可能な並列処理システムに対しては、コンパイラまたはコンパイラプログラムにより、定義ファイルの内容を実行するのに

適した実際のハードウェア構成情報を出力することができる。

【0024】

定義ファイルに基づいて並列処理システムを生成する際には、幾つかの最適化を施すことができる。並列記述により規定された並列処理が、他の並列記述により規定された並列処理の少なくとも一部と同じ共通処理を含む場合は、第1の工程では、共通処理に同一の要素を含む共通構成を割り当て、第2の工程では、共通構成に含まれない要素に対して遅延要素を加える。これにより、要素や、要素を繋ぐ配線などのハードウェアリソースの消費を抑制できる。

【0025】

並列処理システムを構成する複数種類の演算ユニットは、外部入力により処理を変動する手段を備えていても良い。それに対応して、複数の並列記述には、外部入力により処理が変動する並列処理を記述する並列記述を含められるようにすることが望ましい。ネットワークあるいは回路配線を変更しなくても処理内容を変更することができる並列処理システムの設計が可能となる。ネットワークあるいは回路配線と共に演算ユニットの処理内容を変更することも可能であり、さらにフレキシブルにリコンフィグラブルな並列処理システムの設計が可能となる。

【0026】

本発明の定義ファイルに基づき、並列に動作する複数種類の要素を備えたシステムをシミュレートすることが可能である。定義ファイルに基づき、並列に動作する複数種類の要素を備えたシステムをシミュレートする方法およびシミュレータは、定義ファイルに含まれる複数の並行処理を同期して実行する工程または手段を有し、この実行する工程では、複数の入力データを備えた並列記述を実行する際に、それら複数の入力データに、当該システムに入力されてからのレイテンシーが同一のデータを用いる。また、定義ファイルに基づき、並列に動作する複数種類の要素を備えたシステムをコンピュータによりシミュレートするためのプログラムあるいはプログラム製品も本発明により提供され、CD-ROMなどの適当な記録媒体に記録したり、コンピュータネットワークを介して提供することができる。

【0027】

【発明の実施の形態】

図1に、本発明の定義ファイルを用いてハードウェアを設計する過程を示してある。定義ファイル1はDIDL (Device Independent Description Language) と称されており、コンパイラ2によりハードウェアライブラリ3の情報を参照して、ライブラリ3に格納されたハードウェアを用いたハードウェア構成4に変換される。ハードウェア構成4は、DDDL (Device Dependent Description Language) と称されている。コンパイラ2は汎用的なコンピュータ9を用いて実現されており、DIDL1を解釈してDDDL4を出力する並列システム設計用のプログラム5がインストールされることによりコンパイラとして機能する。したがって、DIDL1は、コンピュータ読み取り可能な記録媒体6、例えばCD-ROMあるいは磁気ディスクなどに記録されて提供される。インターネットなどのコンピュータネットワークによる通信を用いて提供することも可能であり、提供されたDIDL1は、コンピュータ9の一部となる適当な記録媒体に記録されて使用される。

【0028】

図2に、ハードウェアライブラリ3に要素の情報が格納されている並列処理システムの一例を示してある。この並列処理システムは、本出願人の国際出願公開公報WO03/007155号に開示されている再構成可能なプロセッサ (RP、Reconfigurable Processor) である。このRP20は、プログラムなどによって与えられる命令セットに基づきエラー処理を含めた汎用的な処理を行う汎用な構成の基本プロセッサ21と、マトリクス状に配置された演算あるいは論理エレメントにより特定のデータ処理に適合したデータフローあるいは擬似データフローがバリエーションに形成されるAAP (Adaptive Application Processor) 部あるいはAAPユニット (以降ではAAP) 50と、このAAP50からの割り込み処理を制御する割り込み制御部22と、AAP50に作動用のクロック信号を供給するクロック発生部28と、このRP20で提供可能な演算回路のフレキシビリティをさらに向上するためのFPGA部27と、外部に対するデータの入出力を制御するバス制御部29とを備えている。基本プロセッサ21とAAP50は、これらの間でデータを交換可能なデータバス24aと、基本プロセッサ21

からAAP50の構成および動作を制御するための命令バス24bとにより接続されている。また、AAP50から割り込み制御部22に信号線25を介して割り込み信号が供給され、AAP50における処理が終了したり、処理中にエラーが発生したときはAAP50の状態を基本プロセッサ21にフィードバックできるようにになっている。

【0029】

AAP50とFPGA27との間もデータバス26により接続されており、AAP50からFPGA27にデータを供給して処理を行い、その結果をAAP50に返せるようになっている。さらに、AAP50は、ロードバス23aおよびストアバス23bによってバス制御ユニット29と接続されており、RP20の外部のデータバスとの間でデータを交換できるようになっている。

【0030】

図3にAAPユニット50の概要を示してある。AAPユニット50は、複数の算術および／または論理演算を行う論理ブロック、論理ユニットあるいは論理要素（以降ではエレメント）がマトリクス状に配置されたマトリクス部51と、そのマトリクス部51に対してデータを供給する入力バッファ52と、マトリクス部51から出力されるデータを格納する出力バッファ53を備えている。これら入力バッファ52および出力バッファ53は、それぞれ4つの小容量の入力メモリにより構成されており、アクセス調停ユニット54を介して入出力バス23aおよび23bに接続される。

【0031】

マトリクス部51が、データバスあるいはデータフローを再構成可能な並列処理システムの中心となる集積回路区画であり、複数種類の演算ユニットであるエレメント55が縦方向に4つのラインを構成するようにアレイ状あるいはマトリクス状に配置されている。このマトリクス部51は、これらのエレメント55の間に配置された、横方向に延びた行配線群57と、縦方向に延びた列配線群58とを備えている。列配線群58は、列方向に並んだ演算ユニット55の左右に分かれて配置された配線群58xおよび58yが1対になっている。行配線群57および列配線群58との交点にはスイッチングユニット59が配置されており、

行配線群 57 の任意のチャンネルを、列配線群 58 の任意のチャンネルに切り替えて接続できるようになっている。各々のスイッチングユニット 59 は、設定を記憶するコンフィグレーション RAM を備えており、プロセッサ部 21 から供給されるデータによりコンフィグレーション RAM の内容を書き換えることにより、行配線群 57 と列配線群 58 の接続を動的に任意に制御できる。このため、このマトリクス部 51 においては、複数のエレメント 55 の全部あるいは一部が配線群 57 および 58 により接続されて形成されるデータフローの構成を任意に動的に変更することができる。

【0032】

RP20 においては、エレメント 55 が並列に動作する要素であり、各種類のエレメント 55 の機能、遅延、入出力データの条件などの情報がハードウェアライブラリ 3 に格納されている。エレメント 55 は、クロック発生部 28 から供給されるクロック信号に同期して稼動するので、ハードウェアライブラリ 3 には各種類のエレメント毎に、入力データを処理して出力するために消費されるサイクル数が遅延情報として格納されている。さらに、各種類のエレメント 55 の配置と、配線群 57 および 58、スイッチングユニット 59 の情報もハードウェアライブラリ 3 に格納されており、コンパイラ 2 からは、DIDL1 に定義されたアルゴリズムを実現するための、エレメント 55 の接続情報（データフロー構成）がハードウェア構成情報（DDDL）4 として出力される。このため、DDDL4 に従ってエレメント 55 が配線群 57 および 58 で接続されるようにマトリクス部 51 を制御することにより、DIDL1 に定義されたアルゴリズムをマトリクス部 51 にエレメント 55 により空間的に割り付けることが可能となる。

【0033】

各エレメント 55 は、1 組の列配線群 58 x および 58 y のそれぞれから入力データを選択するための 1 組のセクタ 54 と、選択された入力データに特定の算術および／または論理演算処理を施し、出力データとして行配線群 57 に出力する内部データパス部 56 を備えている。そして、本例のマトリクス部 51 には、各行毎に異なる処理を行うための内部データパス部 56 を備えた種類の異なるエレメント 55 が並んで配置されている。例えば、第 1 行目に配列されたエレメ

ント55は、入力バッファ52からのデータを受信する処理に適したデータパス部(LD)56iを備えている。第2行目に配置されたエレメント55aは、入力バッファ52に外部デバイスからデータを書き込むためのエレメントであり、ブロックロードするためのアドレスを発生するのに適した内部データパスを具備するデータパス部(BLA)56aを備えている。マトリクス51を構成する全てのエレメント55は、内部データパスの構成あるいは初期値などがある程度変更できるようになっており、その設定は各々のエレメント55のコンフィグレーションRAMに基本プロセッサ21から制御信号24bにより指示される。

【0034】

第3行目に配置されたエレメント55bは、入力RAMの各々より所望のデータをマトリクス部51へロードする入力読み出しアドレスを発生するデータパス部(LDA)56bを備えている。第4行目および第5行目に配列されたエレメント55cは、算術演算および論理演算に適したデータパス部(SMA)56cを備えている。このデータパス部56cは、たとえば、シフト回路、マスク回路、論理演算ユニットALUおよびALUで処理する演算をセットするコンフィグレーションRAMを備えている。したがって、プロセッサ21が書き込んだ命令により、マトリクス部51へ入力されたデータを加算あるいは減算したり、比較したり、論理和あるいは論理積を取ったりすることができ、その結果がエレメント55の出力信号として出力される。

【0035】

その下の行に配列されたエレメント55dは、データが伝送されるタイミングを遅延する処理に適したデータパス部(DE L)56dを備えている。その下の行に配列されたエレメント55eは、乗算器などを含む乗算処理に適したデータパス部(MUL)56eを備えている。さらに異なるエレメント55fとしては、マトリクス部51の外部に用意されたFPGA27とのインターフェイス用のデータパス部56fを備えたエレメントも用意されており、データをいったんFPGA27に供給して処理した後、再びマトリクス部51に戻して処理を継続することができる。

【0036】

これらの再構成可能な集積回路区画 51 のさらに下方には、ストア用のアドレスを発生するのに適したデータパス部 56 g および 56 h をそれぞれ備えたエレメント 55 g および 55 h が配置されている。これらは、出力バッファ 53 を介して外部デバイスにデータを出力するための制御を行う。そして、最下段には、ストア用にデータを出力するのに適したデータパス部 (ST) 56 s を備えたエレメント 55 が配列されている。したがって、マトリクス部 51 を用いて、エレメント 55 の接続を動的に変更することにより、様々なデータフローをフレキシブルに構成でき、様々な処理を行うことができる。

【0037】

図 4 (a) および図 4 (b) に、DIDL の簡単な例を示してある。図 4 (a) に示した DIDL 10 a は 2 行の並列記述 11 a および 11 b を有し、変数 a に変数 b を代入する処理 12 a と、変数 c に変数 a を代入する処理 12 b とが、並列に動作する要素により同期して独立に行われるハードウェア構成を示している。この DIDL 10 a により定義されたハードウェアにおいては、あるサイクル t0 で変数 (a、b、c) が (1、2、3) であれば、次のサイクル t1 で変数 (a、b、c) は (2、2、1) となる。一方、この DIDL 10 a の記載がプログラム 19 a だと理解すると、変数 c は 2 になるので、得られる結果は異なる。しかしながら、サイクル t1 に続く次のサイクル t2 において、DIDL 10 a により定義されたハードウェアにおいては、変数 (a、b、c) は (2、2、2) となり、DIDL 10 a の記載がプログラム 19 a であると理解した場合と同じ結果が得られる。

【0038】

図 4 (b) に示した DIDL 10 a' は、並列記述 11 a および 11 b の順番が入れ替わっているが、それぞれの記述に対応する処理 12 a および 12 b が独立して行われるので各サイクル t0 ~ t2 における演算結果は変わらない。これに対し、プログラム 19 b であると理解すると、処理 12 a と 12 b の順番が入れ替わるので、変数 (a、b、c) は (2、2、1) となる。すなわち、処理が入れ替わったプログラムの結果は、DIDL 10 a または 10 b のサイクル t1 における値に合致する。したがって、DIDL 10 a により、プログラム 19 a

および 19b のいずれのアルゴリズムも実現するハードウェアを記述できるが、プログラム 19a および 19b と同一の結果が得られるサイクルが異なる。そこで、本発明においては、少なくとも 1 つの他の並列処理の出力データを少なくとも 1 つの入力データとする並列処理を示す並列記述によりアルゴリズムを並列システムに割り付けると共に、複数の入力データを備えた並列記述においては、それら複数の入力データは、システムに入力されてからのレイテンシーが同一であると規定することにより、プログラムと同一のアルゴリズムにより演算が可能なハードウェア構成を定義できるようにしている。

【0039】

図 5 に、DIDL 用のコンパイラ 9 の概略処理を示してある。まず、ステップ 31 で DIDL 1 を読み込み、ステップ 32 で DIDL 1 に含まれた並列記述を解釈する。そして、ハードウェアライブラリ 3 に格納されたハードウェア構成に基づき、並列記述に示された並列処理を行う回路構成を生成する。図 3 に示した再構成可能なプロセッサのマトリクス部 51 により並列処理を行う場合は、ハードウェアライブラリ 3 には、各エレメント 55 の情報と、配線 57 および 58 とスイッチ 59 の情報とが格納されており、それらのハードウェア情報にしたがって回路構成する。各エレメント 55 の情報としては、例えば、演算機能、入力条件、処理サイクルがある。マトリクス部 51 では、各エレメント 55 の個数および配置は決まっているので、並列処理を行う回路構成としては、並列処理のために選択されたエレメントと、その位置および選択されたエレメントを接続する配線ルートといった回路情報が生成される。

【0040】

次に、ステップ 33 において、ステップ 32 で生成された回路構成に複数のデータが入力される場合は、それら複数の入力データの、並列処理システム、すなわちマトリクス部 51 に入力されてからのレイテンシーが一致するように遅延要素となるデータパス部 (DEL) 56d を備えたエレメント 55d を加える。さらに、ステップ 32 および 33 では、幾つかの最適化が行われる。DIDL 1 に含まれる並列記述により規定された並列処理が、他の並列記述により規定された並列処理の少なくとも一部と同じ共通処理を含む場合は、ステップ 32 では、共

通処理に同一の要素 55 を含む共通構成を割り当て、ステップ 33 では、共通構成に含まれない要素 55 に対して必要であれば遅延要素 55 d を加えてレイテンシーを調整する。

【0041】

そして、ステップ 34 において、ステップ 33 で生成された回路構成とそれらを接続する情報を含むハードウェア構成情報を出力する。これらのステップは、DIDL1 に含まれる並列記述を個別に読み込んで行うことも可能であり、DIDL1 に含まれる全てあるいは一部の並列記述を読み込んで行うことも可能である。

【0042】

これらのステップ 31 ~ 34 を含む並列処理システムの生成方法は、ステップ 31 ~ 34 の各工程を汎用コンピュータ 9 で実行可能なコンパイラプログラムあるいはプログラム製品 5 として適当な記録媒体に記録して提供することができる。また、ネットワークを介してプログラム製品を提供することも可能である。そして、汎用コンピュータ 9 にプログラム 5 をインストールすることにより汎用コンピュータをコンパイラ 2 として使用できる。したがって、コンパイラプログラム 5 をインストールされたコンピュータ 9 は、図 6 に示すように、DIDL1 を読み込む機能 35 と、ハードウェアライブラリ 3 に基づき、DIDL1 に記述された並列処理を実行するための回路構成を生成する機能（第 1 の機能）36 と、回路構成に複数のデータが入力される場合は、それら複数の入力データの、マトリクス部 51 に入力されてからのレイテンシーが一致するように遅延要素 55 d を加える機能（第 2 の機能）37 と、生成された回路構成を、要素間の接続情報を含めてハードウェア構成情報 4 として出力する機能 38 を備えたコンパイラ 2 として動作する。

【0043】

図 7 (a) に、異なる DIDL の例を示してある。この DIDL10b には、システム、本例ではマトリクス部 51 に入力される変数を示す記述 11c と、内部変数を示す記述 11d と、足し算を示す記述 11e とが含まれている。この DIDL10b がコンパイラ 2 に読み込まれて処理されると、図 7 (b) に示すよ

うに、算術演算が可能なデータパス 56c を備えた演算エレメント 55c を有し、その演算エレメント 55c に変数 b と c が入力され、変数 a が出力される回路構成 18b が生成される。

【0044】

図 8 (a) に、さらに異なる DIDL の例を示してある。この DIDL 10c には、変数に関する記述 11c および 11d に加え、2つの並列処理 12f および 12g を示す並列記述 11f および 11g が含まれている。この DIDL 10c がコンパイラ 2 に読み込まれると、まずは、ステップ 33 の回路構成を生成する機能において演算エレメント 55c を用いた回路構成が生成される。しかしながら、並列処理 12f および 12g は共通した処理を含んでいるので、その部分は共通構成 17c とされ、図 8 (b) に示すように、並列処理 12g は並列処理 12g' に最適化される。その結果、並列処理 12g' においては、演算エレメント 55c に入力される変数は、システム内部で演算処理され、システムに入力される変数 b および c に対して演算エレメント 55c の処理サイクル分だけ後れる変数 a と、システムに入力される変数 d となる。したがって、ステップ 34 のレイテンシーを調整する機能において、入力される変数 a と変数 d とのレイテンシーを調整するために、システム入力される変数 d を演算エレメント 55c のサイクル数分だけ遅らせる遅延エレメント 55d が挿入される。演算エレメント 55c で足し算を行った場合に消費されるサイクル数は、ハードウェアライブラリ 3 に格納されているので、そのサイクル数を消費するように遅延エレメント 55d は設定される。その結果、図 8 (c) に示す回路構成 18c が生成され、複数のエレメント 55 を接続するハードウェア構成情報としてコンパイラ 2 から出力される。なお、以降では、説明を簡単にするために、特に記載しないかぎり、エレメントにおいては 1 サイクルが処理されるものとして説明する。

【0045】

図 9 (a) に、さらに異なる DIDL の例を示してある。この DIDL 10d には、変数に関する記述 11c および 11d に加え、4つの並列処理 12h ~ 12k を示す並列記述 11h ~ 11k が含まれている。この DIDL 10c がコンパイラ 2 に読み込まれると、まずは、ステップ 33 の回路構成を生成する機能に

において演算エレメント 55 c を用いた回路構成が生成される。並列処理 12 k においては、入力される変数 c は、システム入力変数 a を入力とする並列処理 12 の出力である。また、入力される変数 e は、システム入力変数 a および b を入力とする並列処理 12 i の出力をさらに入力とする並列処理 12 j の出力である。したがって、ステップ 34 のレイテンシーを調整する機能において、入力される変数 c と変数 e とのレイテンシーを調整するために、変数 c を遅らせる遅延エレメント 55 d が挿入される。その結果、図 9 (b) に示す回路構成 18 d が生成され、複数のエレメント 55 を接続するハードウェア構成情報としてコンパイラ 2 から出力される。

【0046】

図 9 (a) の記述をプログラム 19 d として考えた場合、処理 12 h ~ 12 k は時間軸に従って上から順番に行われる。したがって、処理 12 k で演算される変数 c および変数 e は、それぞれ、先行する処理 12 h および 12 j によりそれぞれ決定されたものとなる。しかしながら、図 9 (a) の記述を単に並列実行される複数の処理の記述であると考え、並列処理 12 k の変数 c および変数 e は、1 サイクル前に定まった値となり、プログラム 19 d として考えた場合と処理 12 k の出力が異なる。これに対し、図 9 (a) の記述が本発明の定義ファイル、すなわち DIDL 10 d であるとする、並列処理 12 k の変数 c および変数 e は、レイテンシーが同一であると解釈される。したがって、2 サイクル前に確定した変数 c が遅延エレメント 55 d を介して 1 サイクル前に確定した変数 e と同期を取って並列処理 12 k に入力される。その結果、並列処理 12 k の出力 f は、変数 a および b を入力してから 3 サイクル後にプログラム 19 d と同じになり、並列処理を記述した DIDL 10 d によりプログラム 19 d と同じ結果が得られることが分かる。

【0047】

並列処理を記述した DIDL 10 d は、それに含まれる記述 11 h ~ 11 k の順番を変えても処理される内容は同じである。これに対し、プログラム 19 d は、処理がならんだ順番、すなわち、時系列でアルゴリズムを定義しているので、記述を入れ替えると処理される内容は異なる。しかしながら、複数の並列記述に

対して入力変数のレイテンシーが同一になるという定義を導入することにより、プログラムとして記述されたファイルを、並列処理を定義するファイル、すなわち、ハードウェアを定義するファイルとして理解することが可能となる。この結果、プログラマーは、普通に時系列に従ってアルゴリズムを示したプログラムを記述することにより、並列処理システムのハードウェアを記述する本発明の定義ファイルを作成することが可能となる。したがって、本例のDIDL1を用いることにより、プログラマーはプログラムを作成するのと同じ感覚でハードウェア設計を行うことができる。

【0048】

さらに、定義ファイルでは、並列処理に入力される変数のレイテンシーが同一と理解されるだけであり特定のハードウェアを前提としない。すなわち、レイテンシーが同一の入力変数は、同期して、あるシステム（本例ではマトリクスユニット51）にロードされたデータ群に含まれるものか、またはその同期してロードされたデータ群のいずれかが並列処理により処理されたデータ群に含まれるものである、同期してシステムに入力された変数由来であること以外は、実際のハードウェアを前提として設計するとき以外に理解する必要がない。したがって、本発明のDIDLは、ハードウェアに依存しないハードウェア記述言語であると言えることができる。すなわち、どのようなハードウェアを前提としても解釈することができ、ハードウェアが特定されれば、そのハードウェアに定義ファイルに記述されたアルゴリズムを割り付けることができる。したがって、本例のDIDL1は、時間的ではなく、空間的にアルゴリズムを展開できる言語ということができる。このため、プログラムで記述できるアルゴリズムは、本例のDIDL1を用いることにより、すべて、並列に動作する複数の要素、本例であればエレメント55を組み合わせた回路により実行させることができ、DIDL1により、実行するためのハードウェア構成を記述することができる。

【0049】

図10に、さらに異なるDIDLの例を示してある。このDIDL10eには、外部から与えられる所定の数（numOfData）の入力変数inの最大値aを検索する処理が記述されている。初期値をセットする記述111に続く部分

が並列処理を記述した部分であり、入力変数 i_n とそれまでの最大値 a とを比較して最大値をセットする処理 $12m$ の記述 $11m$ と、カウンターを進める処理 $12n$ の記述 $11n$ と、処理 $12n$ のカウントアップした値が所定の数 $num\ of\ Data$ に達したら処理 $12m$ の最大値 a を出力する処理 $12o$ の記述 $11o$ とを備えている。したがって、処理 $12o$ において、システムへ入力される変数 i_n に対してレイテンシーが調整される。

【0050】

図11に、この $DIDL10e$ がコンパイルされたハードウェア構成 $18e$ を示してある。ステップ33の回路構成を生成する機能において、処理 $12m$ は、演算エレメント $55c$ を用いて回路構成され、処理 $12n$ は、アドレス生成用のエレメント $55b$ を用いて回路構成され、処理 $12o$ は2つの演算エレメント $55c$ を用いて回路構成される。処理 $12m$ を行う演算エレメント $55c$ において消費されるサイクル数が、処理 $12n$ のカウント処理を行うエレメント $55b$ において消費されるサイクル数より多いので、遅延エレメント $55d$ が処理 $12o$ のカウンター値を入力する側に追加される。これにより、処理 $12o$ に入力される、処理 $12m$ の出力と処理 $12n$ の出力のレイテンシーが調整される。なお、処理 $12o$ の出力側に配置された2つの遅延エレメント $55d$ は、マトリクス部 51 が3つのセグメントに分かれており、最初のセグメントで処理 $12m \sim 12o$ の回路が構成されるので、他のセグメントを通過して出力用のエレメント $56s$ にデータを転送するためのものである。

【0051】

図12に、図3に示したマトリクスユニット 51 に配置されたエレメント 55 に図11に示した回路構成を割り付けた状態を示してある。本例においては、エレメント 55 が既にマトリクス状に配置された再構成可能なプロセッサ 20 に $DIDL10e$ のアルゴリズムを割り当てている。したがって、図11に示したハードウェア構成の情報は、図12に示したように、選択されたエレメントの配置と、それらを接続する配線ルートの情報としてコンパイラ2で生成され、 $DDDL4$ として出力される。なお、アドレスを生成するエレメント $55a$ および $55b$ は、入力変数 i_n を外部から入力するために使用されている。また、本図には

示していないが、出力用のアドレスを生成するエレメント 55 g および 55 h も同様に使用されている。

【0052】

本例の並列処理システム 20 は、エレメント 55 がマトリクス状に配置された再構成可能なプロセッサ（集積回路装置）であり、各エレメント 55 は、上述したように、ALU などのある程度の規模の演算機能を備え、データを 8 ビット、16 ビットあるいは 32 ビットなどのバイトあるいはワード単位で、特定の目的のために処理するのに適した演算ユニットである。そして、入力データと出力データとをフリップフロップなどを使ってラッチし、クロック信号で同期している。すなわち、各エレメント 55 における入力と出力はクロックで同期されている。したがって、各エレメント 55 で消費されるサイクルは予め予想することができる。さらに、各エレメントは特定の処理を行うのに適したある程度の規模の演算機能を備えているので、DIDL1 に記述された並列処理にエレメントの単位でハードウェアを割り当てて回路構成することができる。したがって、本例の並列処理システム 20 を前提として DIDL1 を解釈すると並列処理で消費されるサイクルは容易に予想でき、回路構成を生成するのが容易であり、さらにレイテンシー調整も容易に行える。このため、ある程度の規模の演算能力を備えて、1 つの並列処理をエレメント単位ではほぼ実行できるデータ処理装置 20 は、本発明の定義ファイルからハードウェアを生成するのに適したアーキテクチャであると言える。

【0053】

また、エレメント 55 の単位で並列処理が割り付けられ、サイクル数も管理されるので、DIDL1 で定義されたアプリケーションを実行している途中で、あるエレメント 55 の機能を外部入力により変えても、他の並列処理に予期しない影響を及ぼさずにアプリケーションを実行させることができる。機能を変えたエレメント 55 におけるサイクル数変動するのであれば、それに対応できるように遅延エレメント 55 d を接続しておき、遅延エレメント 55 d の内部のサイクル数を同様に外部入力により変更し、他のエレメント 55 で形成されたデータフローに影響を及ぼさずに特定のエレメント 55 の処理をダイナミックに変更する

ことが可能である。

【0054】

図13に、DIDLレベルのシミュレータ67を示してある。このシミュレータ67は汎用のコンピュータ9にシミュレータ用のプログラム68をインストールすることにより構成される。したがって、本発明の定義ファイルであるDIDL1に基づき、並列に動作する複数種類の要素を備えたシステムをコンピュータ9によりシミュレートするためのプログラムあるいはプログラム製品68も、CD-ROMなどの適当な記録媒体に記録したり、コンピュータネットワークを介して提供することができる。

【0055】

図14にシミュレータ67の概略動作をフローチャートにより示してある。まず、ステップ71でDIDL1を読み込む。次に、ステップ72で、DIDL1に記述された複数の並列処理を同期して実行する。この際、複数の入力データを備えた並列記述により指示された並列処理を実行する場合は、それら複数の入力データに、システムに入力されるデータとして定義されているデータに対するレイテンシーが同一のデータを用いる。ステップ73で終了条件、例えば、所定の回数、並列処理を繰り返して実行したり、DIDL1に記述された並列処理の結果が所定の値に達するなどの条件が成立すると、ステップ74でシミュレーションした結果を出力する。これにより、ハードウェアに依存しないで、DIDL1に記述されたハードウェアの動作をシミュレートすることができる。

【0056】

【発明の効果】

以上に説明したように、本発明においては、同期して独立に行われる複数の並列処理をそれぞれ規定した複数の並列記述を有する定義ファイルであって、複数の入力データを備えた並列記述においては、それら複数の入力データは、システムに入力されてからのレイテンシーが同一であると解釈する定義ファイルを提案している。この定義ファイルは、並列処理を記述するものであり、ハードウェア記述ファイルであると理解することができ、また、そこにはハードウェア自体は表れないのでハードウェア依存性のないハードウェア記述であると言うことがで

きる。さらに、並列記述であるので、プログラムカウンタの必要のない形態でアルゴリズムを定義することができる。

【0057】

したがって、本発明の定義ファイルにより、従来の高級言語と類似した記述ではあるが、時間順序性のない並列記述により、アルゴリズムに含まれる順序を時間的にではなく空間的に記述でき、並列に動作する複数の要素を備えた並列処理システムの生成を短期間で容易に行うことが可能となる。特に、ALUなどのある程度の規模の演算機能を備えた演算ユニットがマトリクス状に配置された並列処理システム、さらには、演算ユニットの接続を変えられる再構成可能なデータ処理システムを設計したり開発したりするのに本発明の定義ファイルは有用である。

【図面の簡単な説明】

【図1】

定義ファイルであるDIDLからハードウェア構成情報であるDDDLを生成する概略構成を示す図である。

【図2】

再構成可能な並列処理システムの概要を示す図である。

【図3】

独立に並列に動作する複数のエレメントがマトリクス状に配置された並列処理システムを示す図である。

【図4】

DIDLの例である。

【図5】

コンパイラの概略処理を示すフローチャートである。

【図6】

コンパイラの概略構成を示すブロック図である。

【図7】

DIDLの例と、それに対応する回路構成を示す図である。

【図8】

DIDLの他の例と、それに対応する回路構成を示す図である。

【図 9】

DIDLのさらに異なる例と、それに対応する回路構成を示す図である。

【図 10】

DIDLのさらに異なる例を示す図である。

【図 11】

図 10 に示す DIDL に対応する回路構成を示す図である。

【図 12】

図 11 に示す回路構成をマトリクスユニットに割り付けた状態を示す図である。

。

【図 13】

DIDL レベルでシミュレートする概略構成を示す図である。

【図 14】

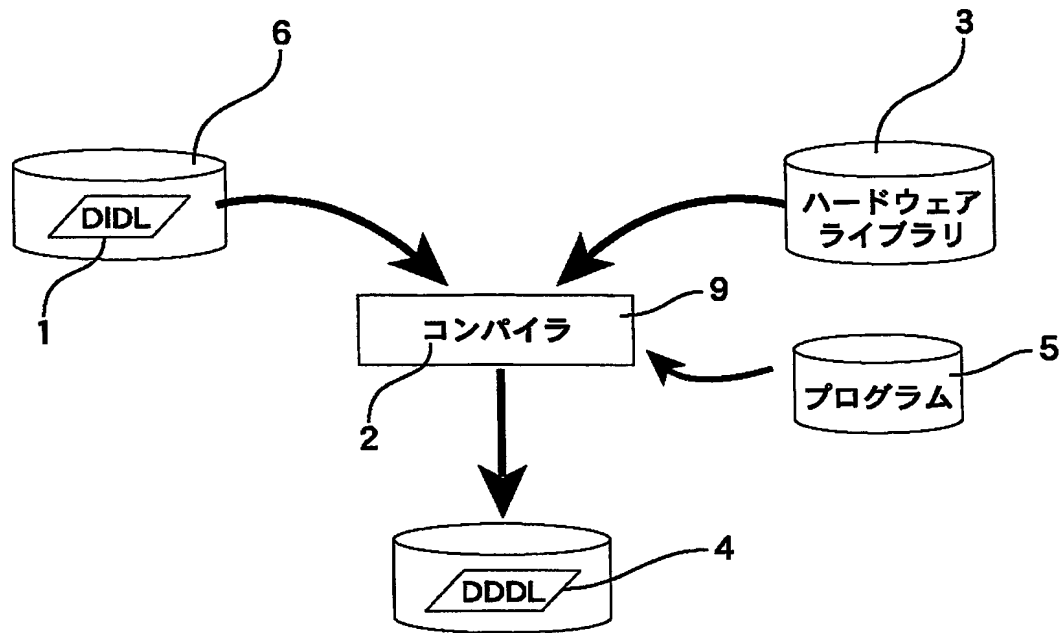
DIDL レベルのシミュレータの処理の概要を示すフローチャートである。

【符号の説明】

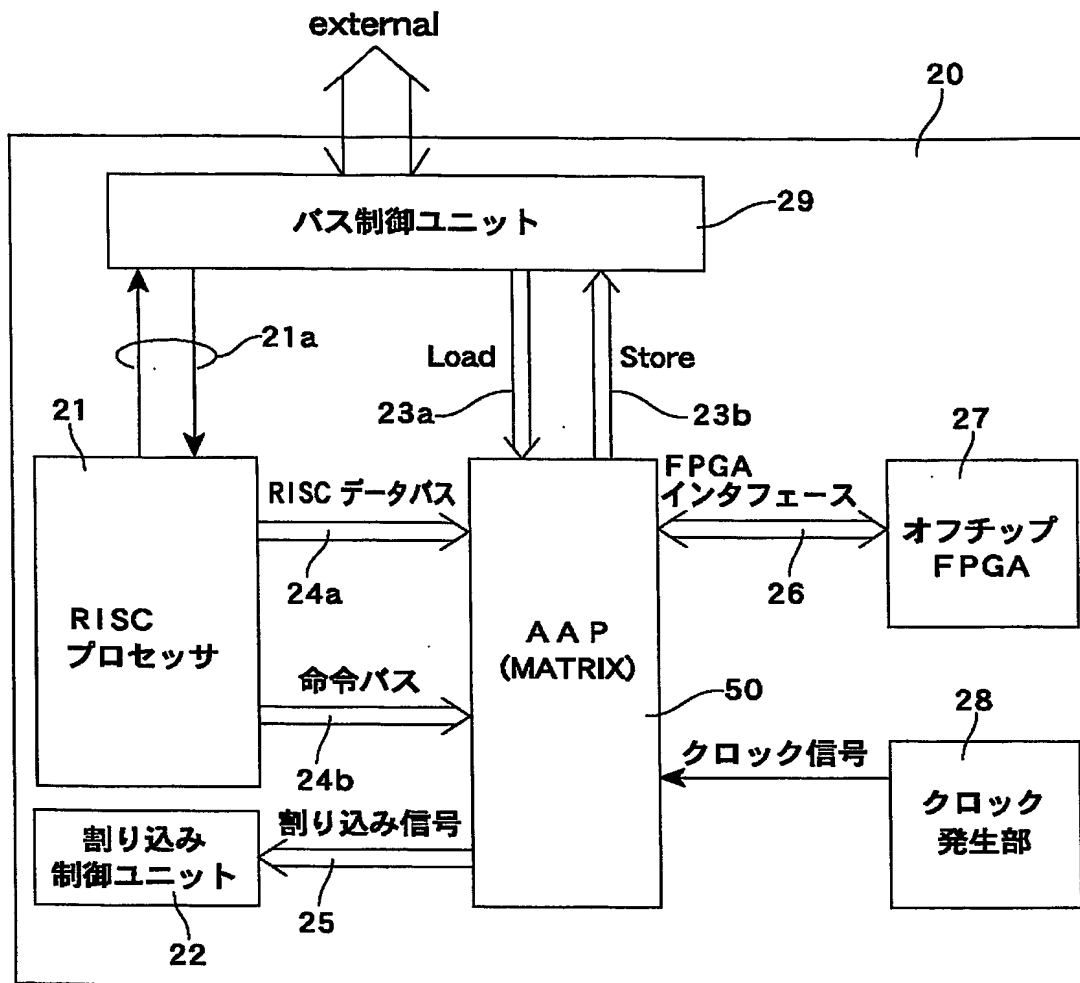
- 1、10a～10e DIDL (定義ファイル)
- 2 コンパイラ
- 3 ハードウェアライブラリ
- 4 DDDL (ハードウェア構成情報)
- 11a～11o 並列記述
- 12a～12o 並列処理
- 20 並列処理システム
- 51 マトリクスユニット
- 55 エレメント

【書類名】 図面

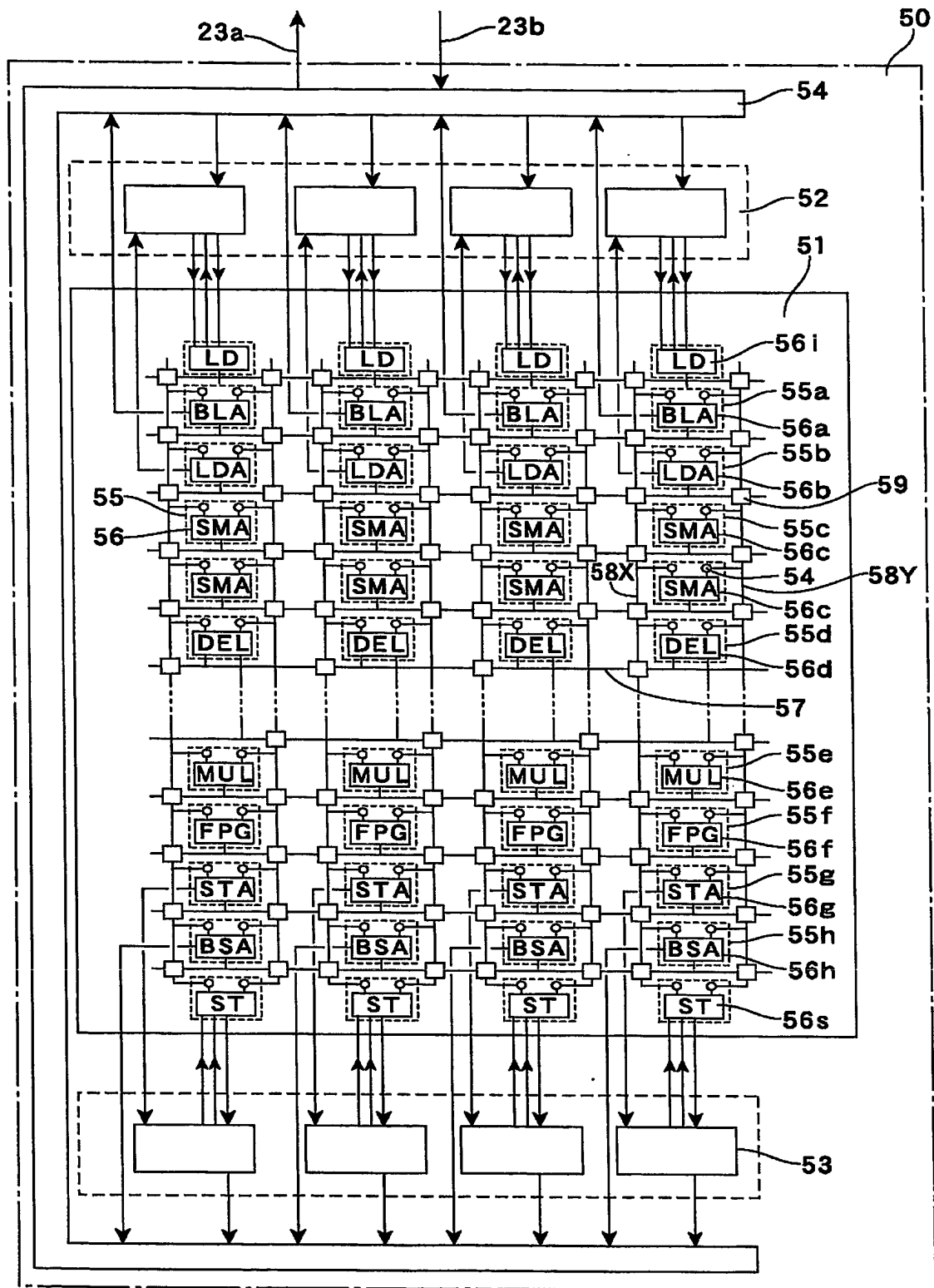
【図 1】



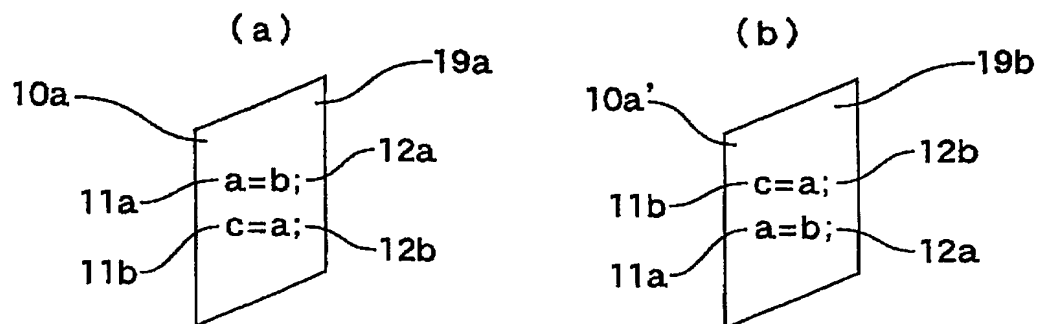
【図 2】



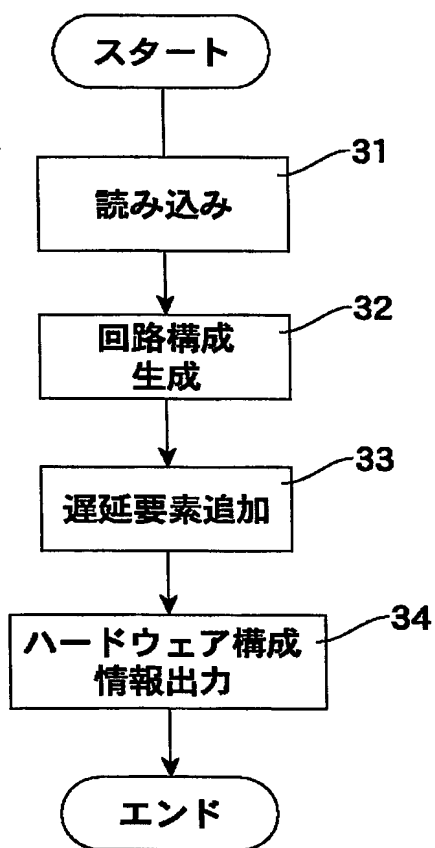
【図 3】



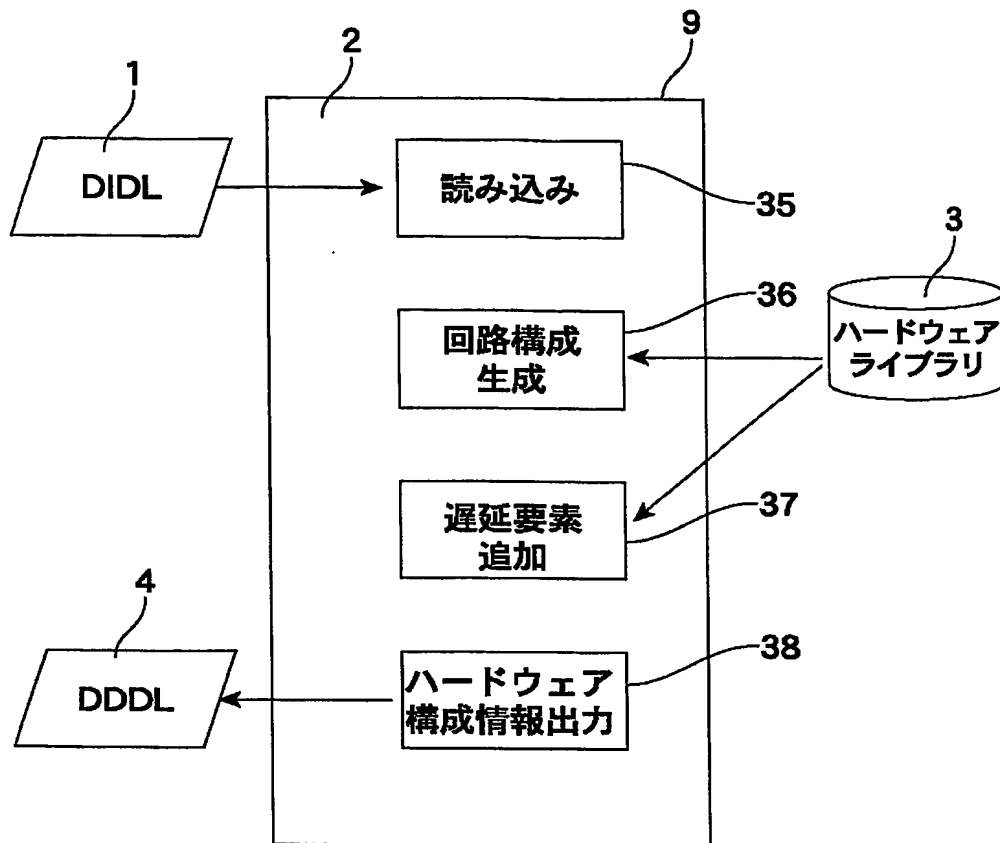
【図 4】



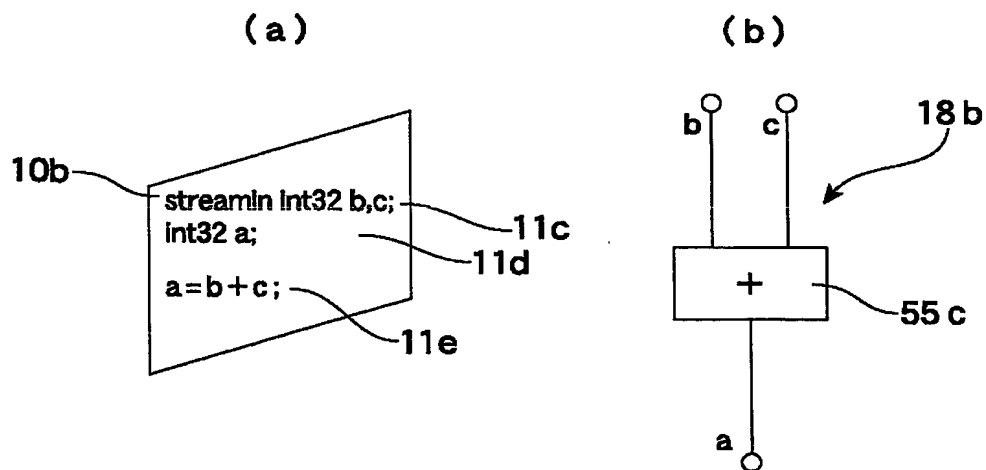
【図 5】



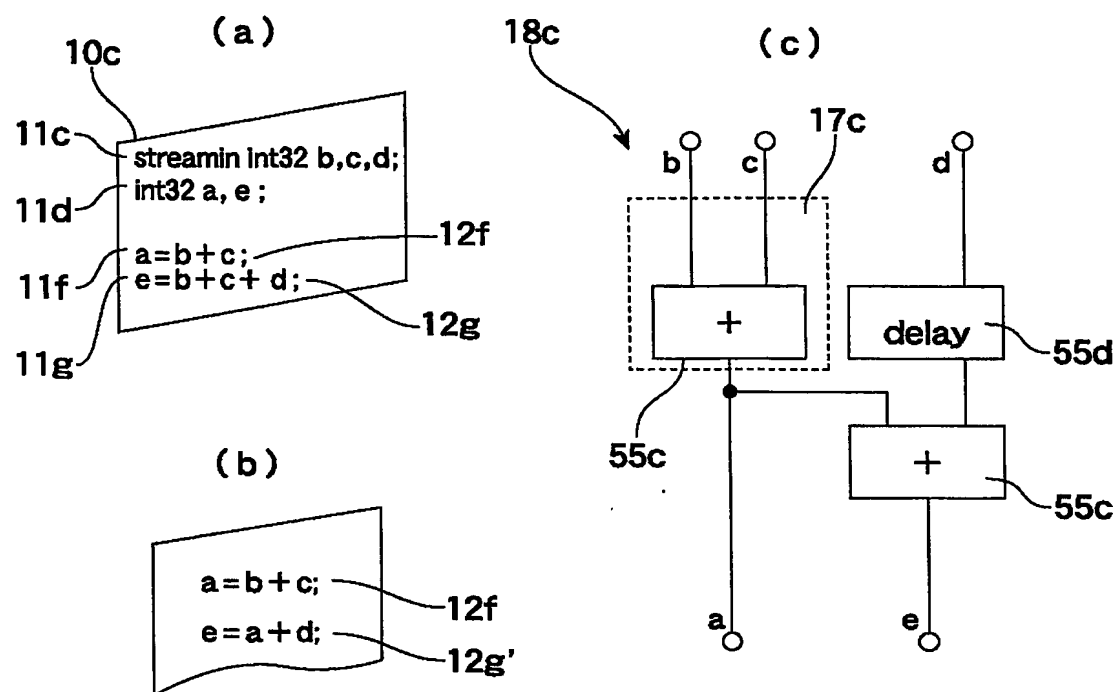
【図 6】



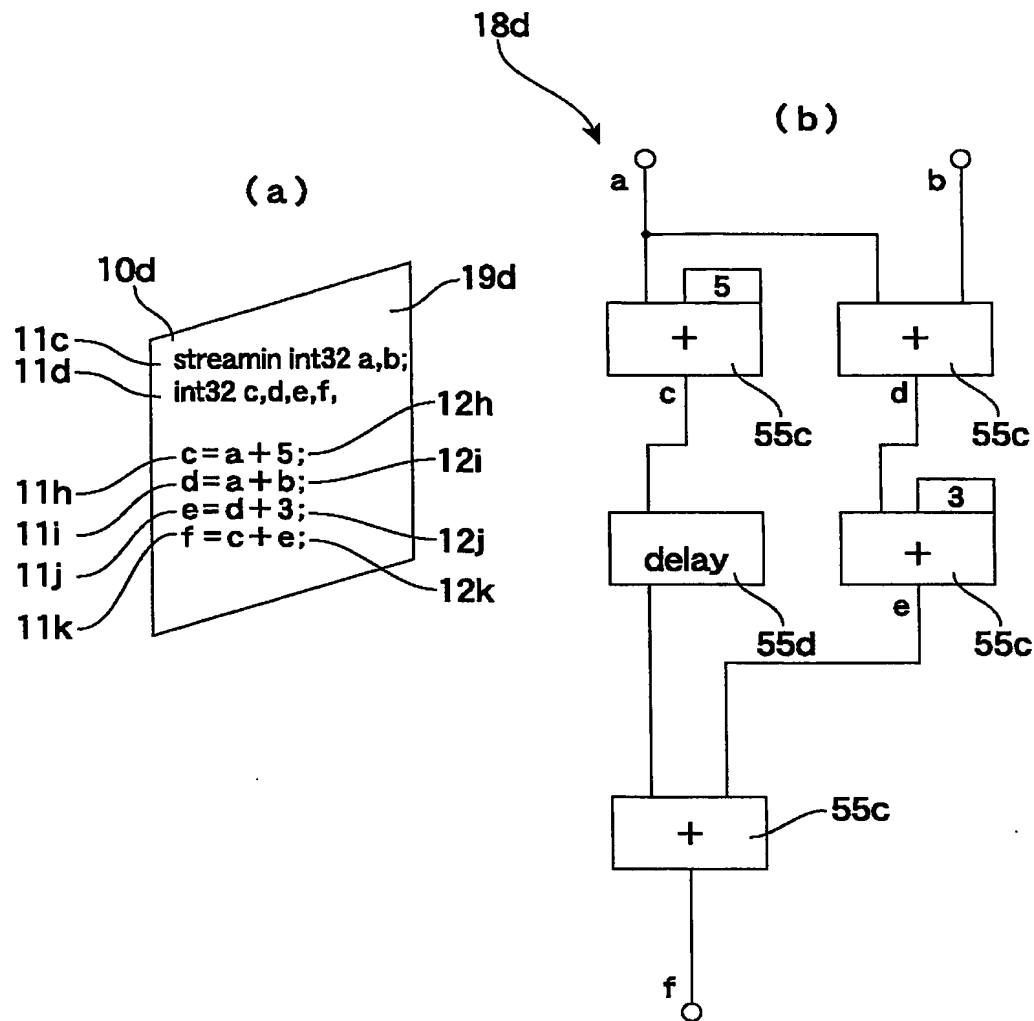
【図 7】



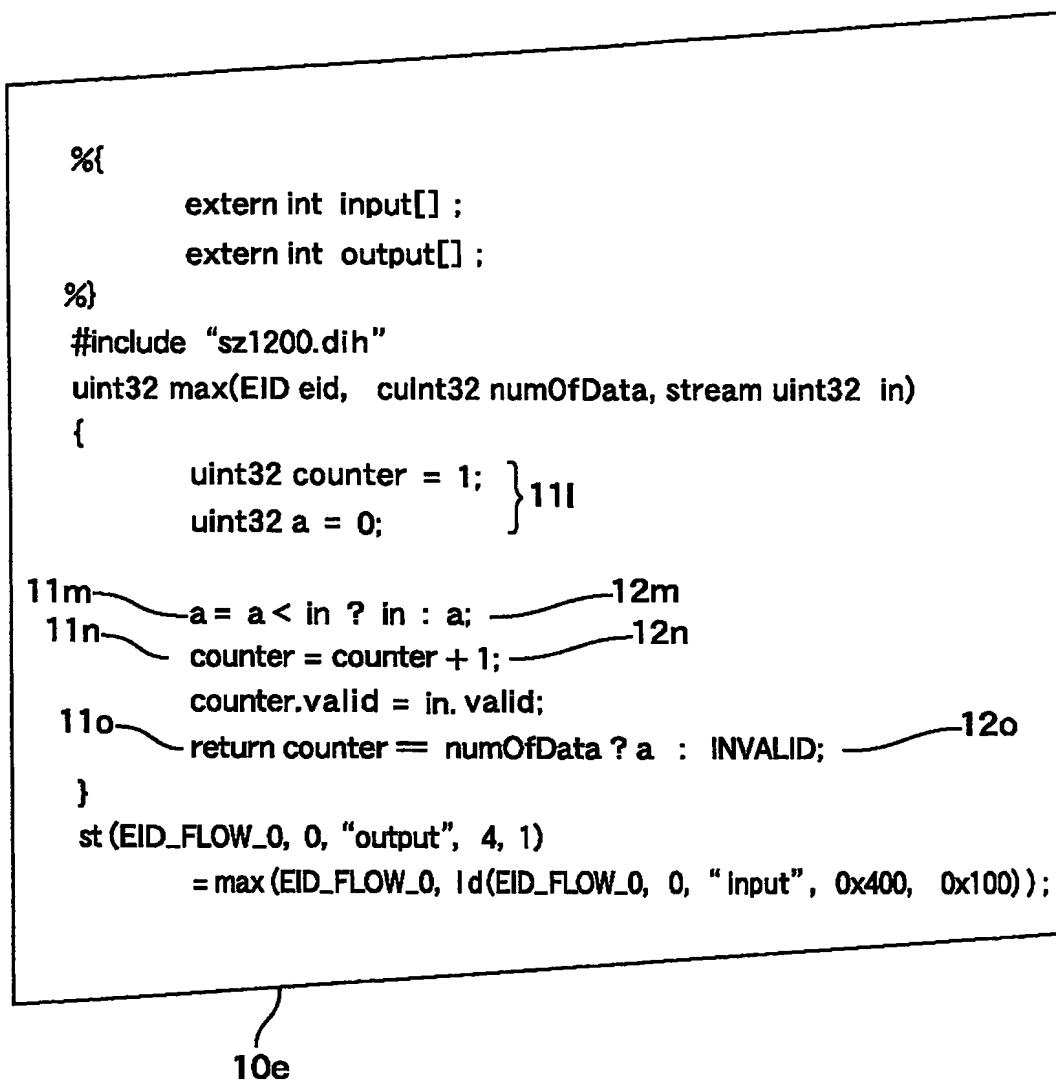
【図 8】



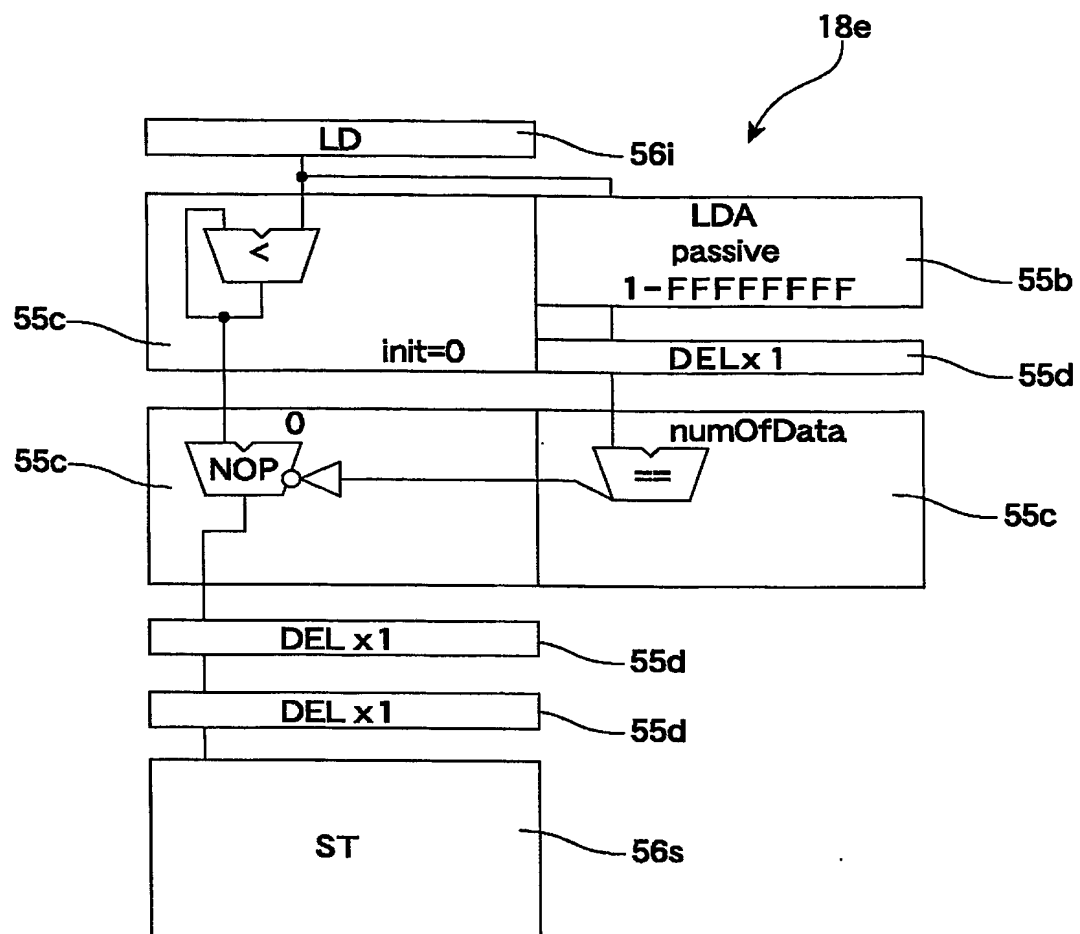
【図 9】



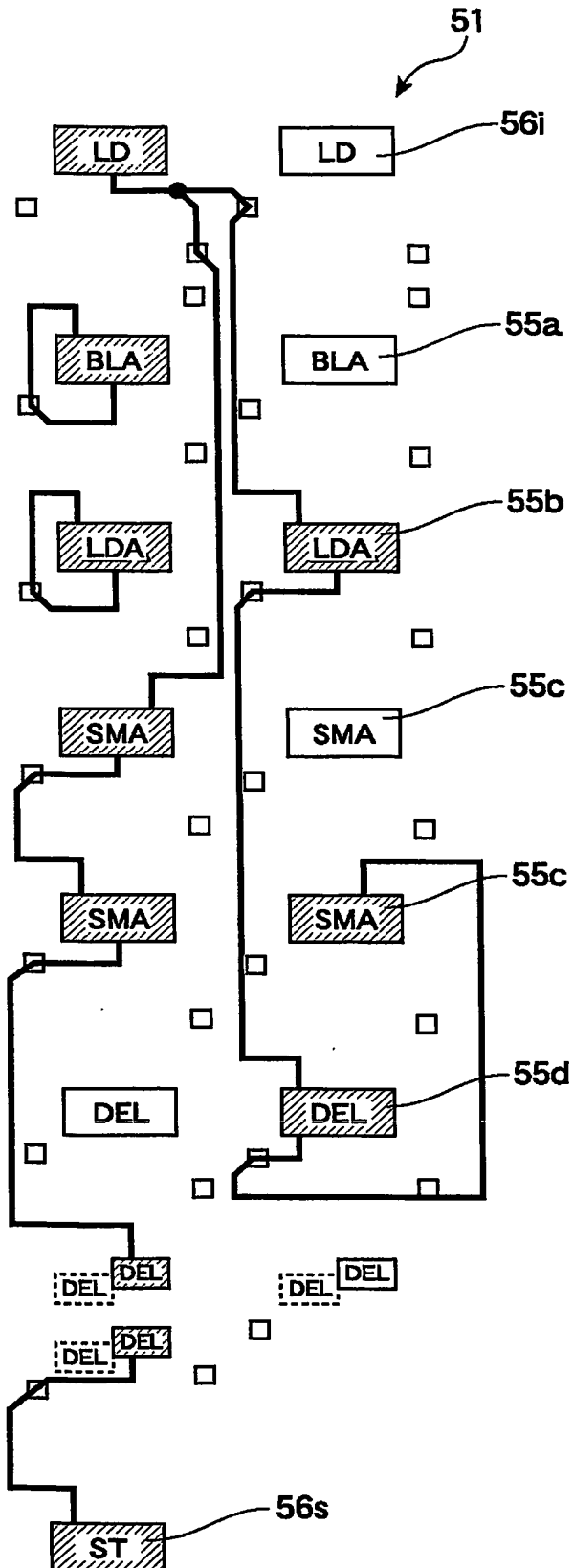
【図 10】



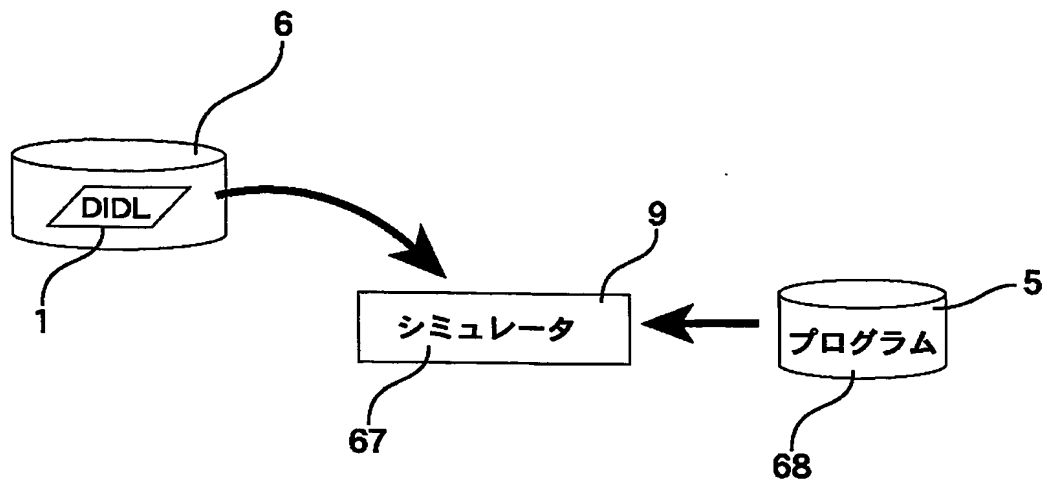
【図 1 1】



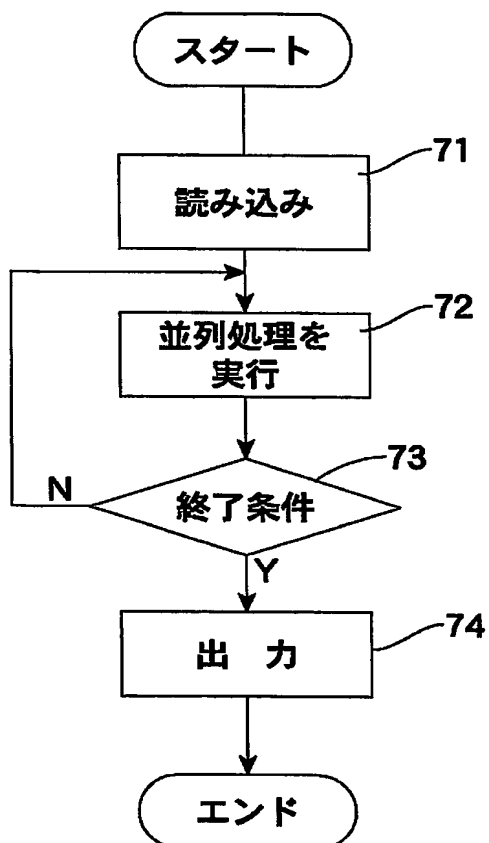
【図 12】



【図 13】



【図 14】



【書類名】 要約書

【要約】

【課題】 並列に独立して動作する複数のエレメントを備えた並列処理システムの開発に適したハードウェア記述手段を提供する。

【解決手段】 同期して独立に行われる複数の並列処理をそれぞれ規定した複数の並列記述を有する定義ファイルであって、複数の入力データを備えた並列記述においては、それら複数の入力データは、システムに入力されてからのレイテンシーが同一であると解釈する定義ファイル（DIDL）1を提供する。コンパイラ2においては、DIDL1を読み込む機能35と、ハードウェアライブラリ3を参照して回路構成を生成する機能36と、入力データのレイテンシーを調整する機能37とを備えており、DIDL1により定義されたハードウェア構成情報を出力することができる。

【選択図】 図6

認定・付加情報

特許出願の番号	特願 2003-185481
受付番号	50301080132
書類名	特許願
担当官	第七担当上席 0096
作成日	平成15年 6月30日

<認定情報・付加情報>

【提出日】 平成15年 6月27日

特願 2003-185481

出 願 人 履 歴 情 報

識別番号

[500238789]

1. 変更年月日

2003年 6月 2日

[変更理由]

住所変更

住 所

東京都品川区上大崎二丁目27番1号

氏 名

アイピーフレックス株式会社